New Tool And Technique For Remote Operating System Fingerprinting

– Full Paper –

Franck Veysset, Olivier Courtay, Olivier Heen, Intranode Research Team

April 2002, v1.1

Abstract – **Information gathering is an essential part of acute vulnerability assessment, especially when the whole process is automated. In this context, host Operating System detection must be precise, even when networks are well defended. We present an original Operating System detection method, based on temporal response analysis. As a proof of concept, we release the open source tool called RING – for Remote Identification Next Generation – and suggest improvements in the paper. We also stress the interesting synergy of using RING together with state-of-the-art tools, such as NMAP [1] or X-Probe [2], for a better overall accuracy in automated vulnerability assessment.**

Index terms – **Remote Operating System Detection, OS Fingerprinting, Automated Vulnerability Assessment, Internet Security.**

1 Introduction

In recent years, the need for automated Internet vulnerability assessment software has been understood and has resulted in the very fast growth of widely available solutions.

As an essential part of the assessment process, remote Operating Systems detection, a.k.a. *OS Fingerprinting*, must meet several requirements:

- **Accuracy**: no falsely detected OS.
- Firewall and IDS **neutrality**: not be disturbed by / do not disturb existing firewalls and IDS.
- **Politeness**: low network traffic and no dangerous segments.
- **Handiness**: easily extensible signature database and automation functions.
- **Speed**: depending on the usage, a fast fingerprinting tool might allow large network scans**.**

We introduce a new OS Fingerprinting method, with such good properties and fairly acute results in practical cases where other tools may fail.

We developed open source software called RING for both *proof of concept* and test purpose. Moreover, we strongly believe that complete access to source code will encourage and speed-up collaborative improvements. RING relies on a signature database that may be enhance, thanks to the built-in learning mode.

2 State-Of-The-Art

2.1 *A brief history of OS Detection*

Security Assessors already have a choice of detection techniques and tools, each of which may be suitable for some application context.

- **Banner grabbing** allows OS deduction from services banner and is appreciated by most human assessors. This can be completed by binary file collect and analysis.
- **TCP segments** (standard or not) response analysis relies on different Operating System responses to specifically prepared segments, particularly when response behavior is not clearly specified in RFCs [3,4]. Furthermore, vendors have introduced fine tuning and proprietary extensions into their TCP/IP stack, which will clearly identify those systems in case of such solicitations. Popular tools such as Savage's QueSO [5] and Fyodor's NMAP[1] [1] use many variants of this technique.
- **ICMP response analysis** is recent. By sending UDP or ICMP solicitation and analyzing various ICMP replies, a tool such as Ofir Arkin's X-Probe [2] will give precise indication except if needed protocols are blocked at firewall level.
- **Initial Sequence Number (ISN) analysis** exploits differences in TCP stacks random generators, identified through a sufficient number of tests [6].
- **Operating System Specific** deny of service are recalled here for the sake of exhaustively, but are only used by hackers. Except for very precise situation, the overall accuracy of this method is rather bad.

For a most comprehensive description of various techniques, see also [7] and Annex 1 – Main Fingerprinting Techniques Comparison.

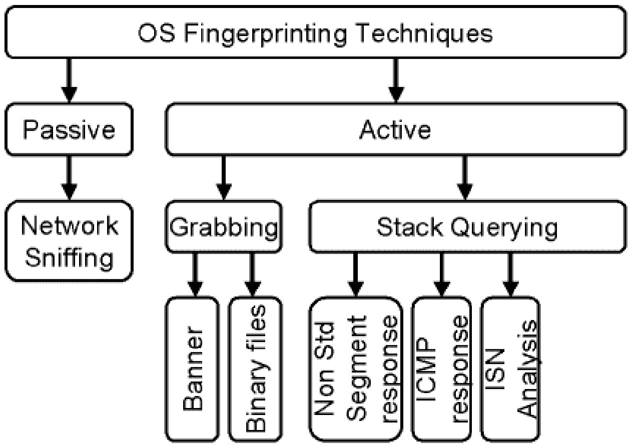


Figure 1: Synoptic of OS fingerprinting technologies

2.2 *Detailing stack querying techniques*

Stack querying techniques allow remote Operating System detection by measuring TCP/IP responses to various solicitations. Most Operating System will answer in a specific manner to specially crafted TCP/IP requests.

Tools such as NMAP or QueSO are based on such techniques. They generate a group of TCP and UDP requests that they send to either opened or closed ports. Then the remote system responses – that can be usual or unusual – are analyzed, providing useful information for eventually deducing the identity of a precise Operating System.

Those techniques generally allow security assessors to get information such as type and version about the target system in a fairly short delay.

Several factors explain the accuracy of the stack querying method:

- Each Operating System or even patch version usually may use its own implementation of the IP stack.

- [1] NMAP has become a *de facto* standard tool, now implementing many techniques, including ISN sampling, ICMP response analysis, UDP probes replies tests...

- TCP/IP specifications are not entirely respected and each different implementation has its own characteristics that can potentially reveal the Operating System.
- Specifications can be interpreted and some features are optional, some constructors implement those features some don't.
- Some proprietary IP improvements are sometimes implemented and are characteristic of an Operating System.

2.3 *Common limitations of classical tools*

NMAP can identify over 500 different Operating Systems, but to do so tests have to be performed in good conditions i.e. on an opened TCP port, a closed one and an closed UDP port. If those requirements are not met the accuracy of the detection will decrease.

With new security policy being used on Internet connected systems, many machines just have one opened TCP port viewable from Internet, every other port being filtered by adequate firewalls or packet filters.

In such a basically secured environment, NMAP, and Xprobe tools, based on ICMP, closed UDP ports, and TCP close ports won't work properly.

3 Needed TCP/IP Material

To make a self-contained paper, we recall some of the most important TCP/IP characteristics.

TCP is a networking protocol whose definition can be found in RFC 793 [3]. ISO norm defines TCP as a data transmission protocol situated over IP. TCP/IP is the main protocol used within the Internet world [9].

Its reliability is the key of its success: Error detection and management, flow and congestion control, duplication control, packet reordering.

To meet this requirements, TCP is connection-oriented. The general scheme is as follows:

1. Connection establishment.
2. Data transfer.
3. Connection Termination.

TCP relies on IP for packet routing over Internet. As network congestion or routing problems can occur, IP can't be trusted for end-to-end packet transmission. Furthermore, IP is a fully connectionless protocol. Thus, connection control has to be performed at the TCP level.

TCP headers contain several fields to manage those features, as show in the table hereafter.

Source Port					**Destination Port**			
Sequence Number								
Acknowledgement Number								
Hlen	**0**	**URG**	**ACK**	**PSH**	**RST**	**SYN**	**FIN**	**Window Size**
Checksum					**Urgent Pointer**			
TCP Options								

Figure 2: Simplified TCP header

The "Sequence Number" and "Acknowledgement Number" fields are used to manage reordering and control particular errors. The URG, ACK, PSH, RST, SYN and FIN fields are used to manage the connection state. RFC 793 defines a state transition diagram for a TCP connection (see Figure 3: Simplified TCP State diagram).

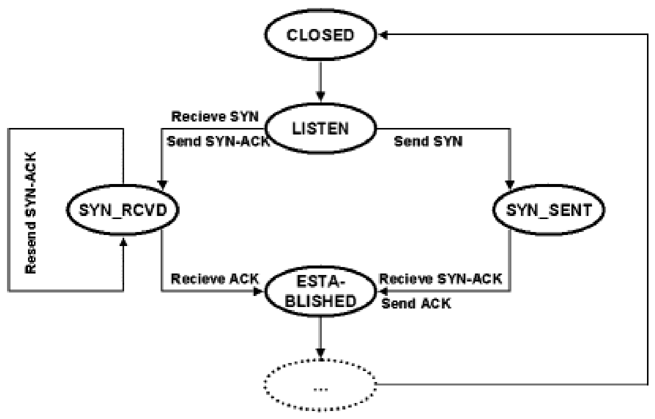


Figure 3: Simplified TCP State diagram

For a better understanding of further explained RING algorithm, it is important to recall the TCP/IP three-way handshake, i.e. the connection establishment method between a client (A) and a server (B).

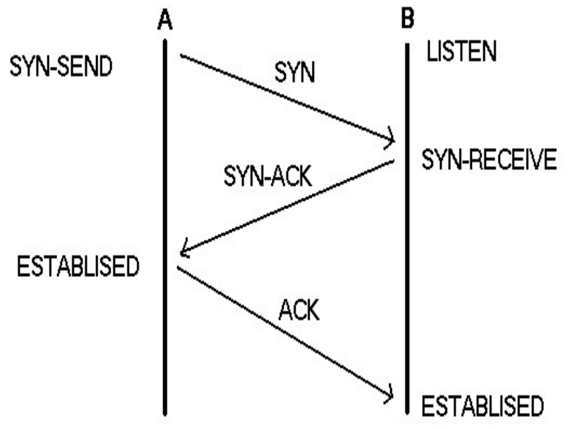


Figure 4: Three-Way Handshake diagram

As some packets might get lost during an IP transmission, "every" packet has to be acknowledged by the receiver. Note that TCP maintains a list of acknowledged packets.

If a packet has not been acknowledged quickly enough to the server, it considers this packet as being lost and resends it.

Moreover, TCP reorders the incoming packets if necessary, so that data is passed in correct sequence to the upper layer.

Network congestion could cause packets to get lost. Any network has a maximum packet per second capacity due to either physical or router performance.

If network congestion occurs, there might be some packet losses. As TCP retransmits lost packets, the congestion problem gets worse and worse. Consequently, if congestion occurs, packet retransmission should be delayed. In other words, the transmission delay in between 2 packets has to increase [9].

This mechanism is specified by TCP, but RFC 793 does not impose any algorithm to compute the acknowledgement delays, it just suggests one.

<u>About the retransmission algorithm (RFC 2988)</u>

TCP is very sensitive to the RTO timer duration:

- Too short: useless retransmission
- Too long: the retransmission comes too late

The protocol has to be efficient for any transmission condition:

- LAN or WAN, heavy or light load
- The timer duration has to be computed according to the Round Trip Time.

RTT evaluation:

- RTT=(α*old_RTT)+((1-α)*measured_RTT), $\alpha \in [0-1]$
- The length of a round trip in between a segment transmission and its acknowledgement.

Segment retransmission and merging make this assessment difficult.

Karn algorithm

While a packet is being retransmitted

RTO = d*old_RTO, d=2

The first implementations proposed

RTO = d*RTT, d=2

The most recent implementations use a variance-based computation

E=((1-j)*old_E) + j*|old_RTT - measuredRTT|)

RTO =RTT + h * E,   h=4, j=1/4

4 Temporal analysis

4.1 Principle description

Packet retransmission offers a pattern that can be analyzed from a remote host.

Such patterns are defined in the norm (RFC 793) but it leaves plenty of scope. Moreover, some of the implementations don't scrupulously respect the norm.

To have a chance of observing these patterns, one must force the target IP stack in a non-standard situation, where timeouts values will be reached.

This can be done by simulating network congestion, simply avoiding to acknowledge the SYN-ACK packets the target emits.

By measuring the delay between packet retransmission, or by looking at some other kind of information such as TCP flags, sequence number or acknowledge number, it is possible to get revealing information about the target behavior.

If every Operating System has its own behavior, it is possible to establish a typical system signature. Whatever the tested machine or the testing conditions, the Operating System is the only element leading the tested machine behavior. Therefore a given test realized on different machines, using the same Operating System, will produce the same result (provide that networks conditions doesn't vary too much).

Comparing the target Operating System fingerprint and the Operating System typical fingerprint, it becomes possible to find out which Operating System is running on the target machine.

New Operating System signature can be easily recorded, then associated to the Operating System name.

Whenever this pattern is observed again it will be easy to recall the learned signature and associated name, which identifies the Operating System.

4.2 Doing it yourself

The Operating System fingerprinting method uses two components of the detector: a packet filter function – such as provided by personal firewalls – together with a packet listening function.

Detector configuration: a simple method to simulate network congestion is to set up a personal firewall on the scanner machine and to create some filtering rules forbidding any incoming traffic from the target machine.

Then a listening mechanism has to be set up for all the packets emitted from the target machine. So the scanner machine TCP/IP stack does not send any SYN/ACK or RST packet that would inform the target machine its packet is received, and thus disable the expected behavior.

Therefore the same machine has to be used as an IP sender, a packet filter and a packet listener.

Test progress: This test progress is composed of three different steps.

- Firewall set up.
- Standard connection attempt on the audited machine.
- Target machine emitted packets monitoring.

Here are the rules to apply to notice retransmission effects: for every following steps the commands are given for Linux 2.4. Note that a packet sniffing tool, such as TCPDump [10], and a command line segment-forging tool, such as SendIP [11] are required.

- Choose a host with a known open port. Let's assume than machine 192.168.1.10 has TCP port 80 open (for example, this system is a web server)
- Configure your firewall to block every incoming packets from the target machine.
  ```
  #> iptables –A INPUT –source 192.168.1.10 –p TCP –sport 80 –dport 62302 –j DROP.
  ```
- Listen to every packets coming from the target and from the open port on the machine. Command line is:

```
#> tcpdump –n host 192.168.1.10
and port 80 and port 62302.
```

- Send a TCP SYN packet to attempt a connection establishment. Command line (on a different shell) is:
```
#> sendip 192.168.1.10 –p TCP –is
your_ip_address –ts 62302 –td 80.
```
- Analyse the delay between every try from the target machine. See Figure 5 for a general scheme of segments transmission. Extensive tcpdump results are indicated below.

```
14:13:12.480412 192.168.1.2.62302 >
192.168.1.10.80: S 221002:221002(0) win 1024
[tos 0x10]

14:13:12.480871 192.168.1.10.80 >
192.168.1.2.62302: S
3566819867:3566819867(0) ack 221003 win 5840
<mss 1460> (DF)

14:13:16.876294 192.168.1.10.80 >
192.168.1.2.62302: S
3566819867:3566819867(0) ack 221003 win 5840
<mss 1460> (DF)

14:13:22.876230 192.168.1.10.80 >
192.168.1.2.62302: S
3566819867:3566819867(0) ack 221003 win 5840
<mss 1460> (DF)

14:13:34.876110 192.168.1.10.80 >
192.168.1.2.62302: S
3566819867:3566819867(0) ack 221003 win 5840
<mss 1460> (DF)

14:13:59.075843 192.168.1.10.80 >
192.168.1.2.62302: S
3566819867:3566819867(0) ack 221003 win 5840
<mss 1460> (DF)1
```

In fact the measure is not based on the time interval in between each retransmission but on the interval in between each packet reception. As the trip time is almost constant, we can assume that those two durations are equal.

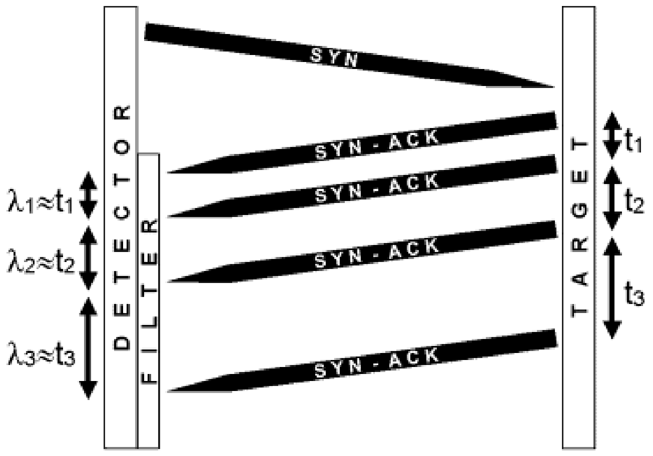


Figure 5: Sending SYN, then ignoring SYN-ACK replies

4.3 Modeling and statistical analysis

Automating this method requires several components:

1. A test and result model.
2. A raw results analysis module.

As the packets may cross an unstable network, such as the Internet, it is likely that the delay between successive packets (λ_i) doesn't exactly equal the delay between these same packets (t_i) at the moment they were sent. See Figure 5.

When two packets are received with a 3.01 second time interval it is very probable that the emission time interval was 3.0 seconds. But some algorithm implementations use 3.2 seconds as a time interval in between packet emissions. The gap between 3.2 and 3.0 is too small for distinguishing one form to the others.

To avoid this problem it is possible to use the TCP Timestamp option, and then gain a better knowledge of when the packet was emitted, at least relatively to the target time reference. Asking for the emission time for every packet on the target machine increases the accuracy of time interval measures. Nevertheless, using timestamps may be intricate as the number given by the TCP timestamp option isn't exactly the date. Rather it is Operating

System dependant increment (generally ranking from 1 ms to 1 second).

This method is based on the reference fingerprints creation and then on a comparison with the experience results. The norm used to measure those distances is the distance in between number series.

Distance = $\Sigma \mid \lambda_i - \delta_i \mid$, λ_i is the time interval associated to the reception time for packet i, δ_i is element i from the fingerprint.

Therefore the qualified OS is the one whose distance is the shortest. This distance does not consider some important packet characteristics such as flags (SYN, ACK, RST, FIN...) used to know the tested machine state or sequence and acknowledgement numbers used to point out some differences between different implementations.

Those features improve the results given by the time measure method by rejecting some fingerprints.

5 RING: implementation And Practical Results

As a proof of concept, we developed the RING tool.

5.1 Libraries

In order to be portable RING has been developed using standard C programming language and some specific UNIX libraries such as Dug Song's Libdnet library [12], Mike D. Schiffman Libnet library [13] and Lawrence Berkeley national laboratory Libpcap library [14].

The Libdnet library is mainly used for firewall control. It provides a development API allowing to control several UNIX firewalls (ipchains, ipfilters, ipf...)

Libpcap is a very common library used to listen and analyze packets on a network without having to use the conventional IP stack.

Libpcap used by RING is taken from NMAP (including Fyodor's modifications).

5.2 Algorithm

The initial arguments needed to perform OS detection with RING are the target host IP address, an opened port on this host, the scanner IP address, and the network interface used to listen to the target responses.

Then RING performs the following internal and network actions:

- Source port choice.
- Using libdnet, set up a local filtering function for blocking every incoming packet from the target machine.
- Using libpcap (pcap descriptor opening), start the packet listening using the filter defined above.
- Using libnet, send a TCP SYN packet to the target machine.
- Listen to the responses for a default or user adjusted delay.
- Compare the obtained responses to the known signatures.

This comparison is based on various parameters: measured values, global duration of measurement, the signature itself and global duration of signature measurement.

Global durations are important. For instance, after 10 seconds it is not possible to distinguish FreeBSD from Windows 2000, the behavior of those two systems are identical. 12 second after the beginning a 3^{rd} SYN-ACK may be received and permit separation between these Operating Systems.

This is the reception or the not-reception of a segment 12 second after the 3rd SYN-ACK that makes it possible to highlight the difference.

Let δi (0<i<n)-n elements of signature-and λj (0<j<m) the m taken measurements. Let T_S duration for the signature generating and T_M duration for generate the fingerprint.

The signature can be rejected if:

- n>m and T_S<T_M
- n<m and T_S>T_M

I.e. like in the signature, it was envisaged the arrival of a packet and that waiting time was sufficiently long (so that it is taken into account), the packet did not arrive. That makes it possible to exclude this signature.

The best signature candidate is the one that minimize the distance $\Sigma \mid \lambda_i - \delta_i \mid$, $(1<i<\min(n,m))$.

Note that RING is available in two different flavors: standalone program and patch for NMAP 2.54BETA30.

5.3 Practical Results

This method offers good accuracy in cases where other tools are tricked. For instance it is possible to differentiate a Win2K from a FreeBSD, even when hosts are hidden behind a commonly configured firewall.

Win2K and FreeBSD implementations have a very similar behavior as they share the same IP stack technology. If there is just one opened port on the tested machine NMAP will confuse them most of the time.

If the technologies are very close, this is not the case for timeouts values, and choices concerning "Reset" packet sending. This is enough for RING to differentiate these Operating Systems, as shown in measure table hereafter.

Retries	Microsoft Windows 2000	FreeBSD 4.4
1st	3	3
2nd	6	6
3rd	No more retries	12
4th		24
Reset	No Reset Sent	Reset after 30 sec.

RING is also capable of distinguishing versions of a same Operating System:

Retries	Linux 2.2.14	Linux 2.4
1st	3,5	4,26
2nd	6,5	6
3rd	12,5	12
4th	24,5	24
5th	48,5	48,2
6th	96,5	No more retries
7th	120,5	
8th	No more retries	
Reset	No Reset	No Reset

Retries	Windows 98	Win 2K
1st	3	3
2nd	6	6
3rd	12	No more retries
4th	No more retries	
Reset	No Reset	No Reset

The following examples show the differences between various equipments:

Retries	Minolta Printer	Cisco Router	3COM Switch
1st	4,5	2	3,5
2nd	4,5	3,9	4,4
3rd	9	5,9	4,4
4th	18	No more retries	4,4
5th	36		4,4
6th	72		4,4
7th	144		4,4
8th	285		4,4
9th	576		4,4
10th	169		4,4
11th	169		4,4
12th	169		4,4
Reset	Reset	No Reset	No Reset

Note that, after a certain number of retries some implementation stop retransmitting by sending a

RST packet to warn the scanner machine the transmission has been broken.

6 Discussion and extensions

6.1 Advantages

The main advantage of RING's method is the use of only one opened port. If the target system is well protected, behind a firewall, chance are that only one port will be open, all other being filtered.

With this kind of configurations, tools such as NMAP are not as efficient because lots of NMAP tests are based on closed ports.

Moreover, the proposed technique uses a standard TCP packet. RING testing won't disturb the machine.

Only few implementations change their TCP state diagram between two versions.

There are only a few differences between Win98 Millennium edition, Windows 2000 and Windows XP. Nevertheless, in order to improve their efficiency the values used by the algorithm are changed more often.

As a matter of fact, the proposed method has a better accuracy than the classical techniques for some implementation.

On the other hand, note that this method takes more time than techniques used by NMAP or Xprobe. This is an inherent drawback as successive times are measured.

6.2 Protection

What are the available protections to escape RING Operating System detection? As the packet sent is standard and unique, it is impossible to distinguish it from any normal traffic on the target machine.

Packet retransmissions are visible but packet loss and retransmission are normal for any network.

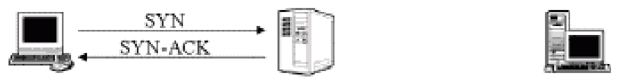
If an IDS aborts a connection in order to prevent too much information from leaving the network it is going to decrease the TCP error recovery capacities.

With some Operating Systems, it is possible to modify some elements in the TCP/IP stack, allowing the system to hide its identity to RING. This method is not advised because it is dangerous for TCP/IP stack stability.

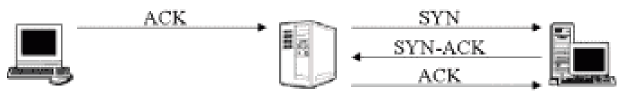
A possible method is to hide the machines behind a proxy or use Firewall that implement SYN Relay or SYN Gateway techniques. SYN Relay or SYN Defender is use to protect host against TCP Flooding.

Thus, the audited stack TCP/IP is the stack of the Firewall instead of the tested host.

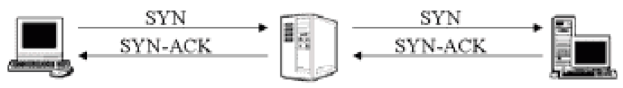
SYN Relay principle



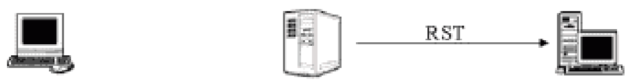
If after a certain time no ACK receive by FW, the connection is aborted. Otherwise, the transmissions continues:



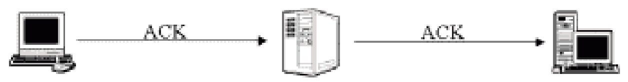
SYN Gateway principle



If after a certain time no ACK receive by FW, the connection is aborted:



Otherwise, the transmissions continues:



6.3 Further improvements

There exist other states in TCP transition diagram that will show similar behavior, trying to reinject supposedly lost segments. This is the case for FIN_WAIT_1 state that can be used to corroborate previous OS deductions and / or bypass some SYN defenders.

FIN_WAIT_1 is a really interesting state as it allows a TCP flag differences analysis. Those differences are similar to the ones used by Nmap, but Nmap can't see them because it is necessary to put the machine in a FIN_WAIT_1 to observe them.

The interesting fact about this method is that it will bypass SYN gateway mechanism, and perform OS fingerprinting tests on the real target.

Retries	Linux 2/4	Win 2K
1st	0,8	3
2nd	1,3	6
3rd	2,6	12
4th	5,2	24
5th	10,5	48
6th	20,8	No more retries
7th	41,6	
8th	No more retries	
Reset	No Reset	No Reset

Further researches on this topic could concern independence in regard to network performance variation and global robustness of RING detection. Indeed, we experiment some instabilities during some measurement sessions. Repetitive measures with some aberrant value detection may help in cases where the network is very unsteady. As indicated in § 4.3, we may use the timestamp TCP field in some cases and then calculate more precise duration between successive segments sending.

Lastly, we feel that known signature file must grow, and guess that open source developer's community will help. We encourage sending comments and newly found signatures to ring@intranode.com

7 Conclusions

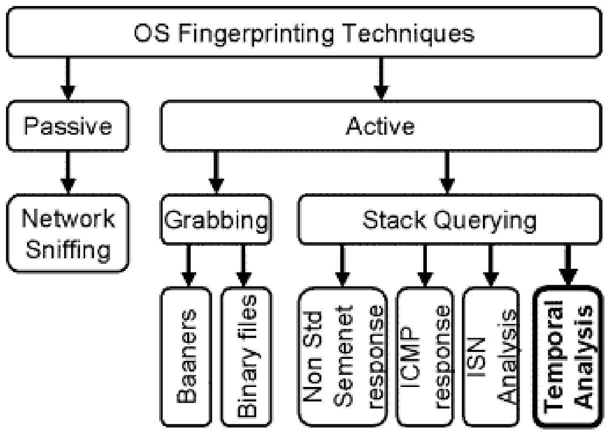


Figure 6: Synoptic of Operating System fingerprinting technologies, now including temporal analysis

RING uses a brand new Operating Systems detection technique, that relies on very common and noiseless TCP traffic. Automated vulnerability assessment engines may greatly benefit form RING, especially when used in conjunction with other techniques (see figure 4).

For further reading and information concerning RING, a full paper can be found at this URL:

www.intranode.com/site/techno/ring-full-paper.pdf

The open source version of RING, with associated libraries, man page and an evolutive signature database can be found at this URL:

www.intranode.com/site/techno/techno_articles.htm

Any comment or suggestion may be sent to the alias ring@intranode.com

8 Acknowledgements

The authors would like to thank Fyodor, D. Fort, P. Auffret, F. Frade, A. Floch and C. Patel for valuable comments.

9 References

[1] Fyodor, *NMAP*, www.insecure.org/nmap

[2] Arkin, O., X-Probe, www.sys-security.com/html/projects/X.html

[3] Postel, J. (Sep, 1981), *RFC 793 – Transmission Control Protocol*

[4] Paxson, V. and Allman, M. (Nov, 2000), *RFC 2988 – TCP Retransmission Timer*

[5] Savage, *QueSO*, savage.apostols.org/projects.html

[6] Zalewski, M. (Apr, 2001), *Strange Attractors and TCP/IP Sequence Number Analysis*

[7] Veysset, F. (Jun, 2001), *13th Annual FIRST Conference – OS Fingerprinting Revisited*

[8] Comer, D and Lin, J. (1994), *Probing TCP implementations* www.cs.purdue.edu/homes/lin/probe.tcp.html

[9] Stevens, W. R. (1994), *TCP/IP Illustrated, Vol. 1*

[10] *TCPDump*, www.tcpdump.org

[11] *SendIP*, freshmeat.net/projects/sendip

[12] *Libdnet*, libdnet.sourceforge.org

[13] *Libnet*,www.packetfactory.net/Projects/Libnet

[14] *Libpcap*, www.tcpdump.org

10 Annex 1 – Main Fingerprinting Techniques Comparison

OS Fingerprinting Techniques Comparison	Banner Grabbing	ICMP replies	Non standard segments	Sequence analysis	Time based
History					
Classical implementation	Plenty	X-PROBE	NMAP	?	RING
Created by	Hackers	Ofir Arkin F. Yarochkin	Fyodor	M. Zalewsky Guardent	Intranode
First released	?	Aug, 2001	Jun, 1998	Apr, 2001	Mar, 2002
IP Protocol & Service					
Used protocols	Service related	UDP & ICMP	IP, TCP, UDP & ICMP	TCP	TCP
Open TCP port requiered	Service related	No	The more the better	Yes	Yes
Closed TCP port requiered	No	No		No	No
Closed UDP port requiered	No	Yes		No	No
Firewall concerns					
Bypass filtering routers	Always	Rarely	Generally	Generally	Generally
Bypass SYN relays	Always	Always	Rarely	Never	Never
Bypass application proxies	Rarely	Never	Never	Never	Never
Outgoing firewall neutral	Always	Generally	Generally	Always	Always
IDS concerns					
Detection	Hard	Possible	Easy	Easy (?)	Hard
Blocking	Hard	Possible	Possible	Possible	Hard
Misc.					
Learning functions	No	No	Yes	?	Yes
KB size (March 2002)	?	~ 20 (?)	> 600	~ 10 (?)	~ 30
Target hosts disturbance	None	None	Rare	Possible	None
Best match feature	No	?	Yes	?	Yes
Defensive measures	Banner rewriting	ICMP blocking at firewall	Firewall or host stack tuning	SYN Relaying	SYN Relaying

11 Annex 2 – Ring Man Page

```
NAME
        ring  0.0.1    -    Remote      Identification    Next
Generation.
        Remotely detects OS types and versions.

SYNOPSIS
        ring [-v][-f fingerprint][–t timeout] –d target_ip
        –s source_ip  –p open_port  –i device

DESCRIPTION
        Ring  performs remote Operating System detection based
on
        temporal analysis of target reemitted SYN-ACK segments.
        measured   values   are  compared  to  reference  values
stored
        in the fingerprint file using a best match algorithm.

OPTIONS
         -d target
                IP address of the tested host
         -s src
                IP address of your host
        -p open_port
                 An open TCP port on the target
         -i device
                the  name  of  your  network  interface  for  reach
                tested host
        -t timeout
                duration for waiting packet
        -v(erbose)
                For analysis and debugging purpose.
        -f(ingerprint) fingerprintfile
                Use alternate signature file.
                default is ./fingerprint


TIPS
         The   same   signature   may   appear   more   than   once   in
a
        signature file, with slightly      different values.

        It    might    be    usefull    to    create    specialized
signature
        files:    one  for  intranet,    one  for  internet  with
possible
        distortion...
```